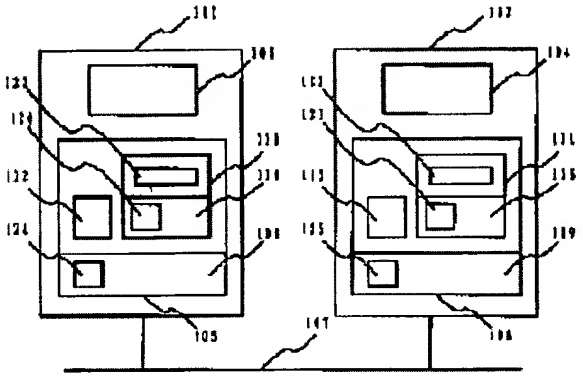
DATA TRANSFER CONTROL METHOD FOR PARALLEL COMPUTER

Publication number: JP9146903
Publication date: 1997-06-06
Inventor: SAGAWA NOBUTOSHI; SUKEGAWA NAONOBU
Applicant: HITACHI LTD
Classification:
- international: ***G06F13/00; G06F15/16;*** **G06F13/00; G06F15/16;** (IPC1-7): G06F15/16; G06F13/00
- european:
Application number: JP19950307111 19951127
Priority number(s): JP19950307111 19951127

Abstract of **JP9146903**

PROBLEM TO BE SOLVED: To transmit and receive messages safely on a parallel computer by using remote memory transfer. SOLUTION: When messages are transmitted to and received from the parallel computer by remote memory transfer, the respective element computers are provided with an arbitrary number of memory areas 122 and 123 that can be accessed remotely and their management areas 124 and 125, and flags for remote access inhibition are provided in the management areas for the respective memory areas, thereby optionally allowing the areas to be accessed remotely or inhibiting them from being accessed remotely. Consequently, data are prevented from being overwritten to the same memory area owing to multiple writing and illegal data are prevented from being read out owing to the repetition of writing to and reading from the same memory area.



(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平9－146903

(43)公開日　平成９年(1997)６月６日

(51)Int.Cl.[6]		識別記号	庁内整理番号	ＦＩ			技術表示箇所
Ｇ０６Ｆ	15/16	４７０		Ｇ０６Ｆ	15/16	４７０Ｄ	
	13/00	３５５			13/00	３５５	

審査請求　未請求　請求項の数３　ＯＬ　（全 11 頁）

(21)出願番号　特願平7－307111

(22)出願日　平成７年(1995)11月27日

(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地

(72)発明者　佐川　暢俊
東京都国分寺市東恋ケ窪１丁目280番地
株式会社日立製作所中央研究所内

(72)発明者　助川　直伸
東京都国分寺市東恋ケ窪１丁目280番地
株式会社日立製作所中央研究所内

(74)代理人　弁理士　小川　勝男

(54)【発明の名称】　並列計算機におけるデータ転送制御方法

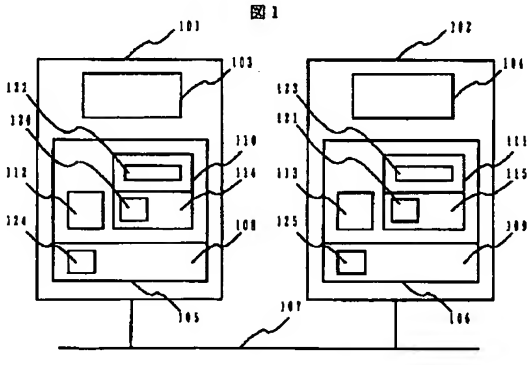
(57)【要約】

【課題】並列計算機上でリモートメモリ転送を用いたメッセージの送受信を安全に行なう。

【解決手段】並列計算機上でリモートメモリ転送によりメッセージを送受信する際に、各要素計算機上にリモートアクセス可能な任意個数のメモリ領域（１２２，１２３）とその管理領域（１２４，１２５）を設け、各々のメモリ領域について管理領域上にリモートアクセス禁止のフラグを設けることにより、その領域に対してリモートアクセスを任意に許可あるいは禁止できるようにする。

【効果】同一メモリ領域への複数の書き込みによりデータがオーバーライトされること、および同一メモリ領域への書き込みと読み出しの重複により不正なデータが読み出されることを防止できる。

図１



1

【特許請求の範囲】
【請求項１】複数台の計算機を通信路によって結合してなり、該計算機がデータ転送先の計算機のメモリ領域を指定してデータ転送を行う並列計算機におけるデータ転送制御方法であって、
各々の計算上で、他の計算機上で実行されているプログラムからデータの読み出し、書き込みを行なうことのできるメモリ領域を、自計算機上で実行されているプログラムから任意個数指定し、
該指定された複数のメモリ領域の各々について、他の計算機上で実行されているプログラムからのデータの読み出しおよび書き込みが禁止されていることを示す情報を保持し、自計算機上で実行されているプログラムから該情報を制御することにより、該メモリ領域へのデータの読みだしおよび書き込みを禁止することを特徴とする並列計算機におけるデータ転送制御方法。
【請求項２】該指定された複数のメモリ領域のうち、他の計算機上で実行されているプログラムからのデータの読み出しおよび書き込みを禁止された該メモリ領域に対し、他の計算機で実行されているプログラムからデータの読み出しが要求された場合、該他の計算機上のプログラムに対しエラーを通知することを特徴とする請求項１記載の並列計算機におけるデータ転送制御方法。
【請求項３】該指定された複数のメモリ領域のうち、他の計算機上で実行されているプログラムからのデータの読み出しおよび書き込みを禁止された該メモリ領域に対し、他の計算機上で実行されているプログラムからデータの書き込みが要求された場合、該データを破棄し、
該データを破棄した計算機において、該データの破棄を、該計算機上で実行されているプログラムに対して通知することを特徴とする請求項１記載の並列計算機におけるデータ転送制御方法。
【発明の詳細な説明】
【０００１】
【発明の属する利用分野】本発明は複数の要素計算機（プロセシングユニット、以下ＰＵ）を通信網によって結合した並列計算機におけるＰＵ間のデータ通信に係わり、特にデータ通信が相手ＰＵの介入なしに行なわれるリモートメモリへのデータ転送時のデータの安全性の確保に関する。
【０００２】
【従来の技術】並列計算機は、複数のＰＵを通信網によって結合し、それらを同時に稼働させることによって処理速度を向上させる。本発明では特に、各ＰＵがそれに付随するメモリ空間のみをアクセスすることができる分散メモリ型の並列計算機を対象とする。分散メモリ型並列計算機では、他のＰＵのメモリ上にあるデータを直接アクセスすることはできない。データが必要となる度に送受信を行なってそのデータを自ＰＵに移動する必要がある。

2

【０００３】分散メモリ型並列計算機では、ＰＵ間のデータのやりとりをすべてプログラム中に記述する必要がある。ここで、ＰＵ間で受け渡されるデータをメッセージと呼ぶ。並列計算機用プログラムでは、他のＰＵで必要となるデータが自ＰＵのメモリ上にある場合にはそれをあらかじめ送信し、他のＰＵのメモリ上にあるデータを自ＰＵが必要とする場合にはあらかじめ受信しておくような指示を各ＰＵのプログラム中に明示的に記述する必要がある。多くの並列計算機システムでは、このようなＰＵ間のメッセージの送受信をサポートする目的で、メッセージパシングライブラリと呼ばれる関数（あるいはサブルーチン）群があらかじめ用意されており、通信は「Ｃ」や「ＦＯＲＴＲＡＮ」などのプログラムからの関数コールとして記述できるようになっている。メッセージパシングライブラリの中には、異なる並列計算機ハードウェア上にインプリメントされ、事実上の標準としての通信環境を提供するものも現われている。米国Ｏａｋ Ｒｉｄｇｅ Ｎａｔｉｏｎａｌ Ｌａｂｏｒａｔｏｒｙで開発されたＰＶＭや、近年標準化が進められているＭＰＩ（Ｍｅｓｓａｇｅ Ｐａｓｓｉｎｇ Ｉｎｔｅｒｆａｃｅ）はその例である。これらの通信ライブラリをコールすることにより書かれた並列プログラムは、異なる並列計算機上でも再コンパイルのみで動作させ得る可能性（可搬性）が高い。
【０００４】通信ライブラリでＰＵ間のメッセージの受け渡しを行なうには、２種類の方式が知られている。第１は、送信側ＰＵでメッセージの送信関数をコールし、受信側ＰＵでそれに対応するメッセージの受信関数をコールし、これらの間でメッセージを送受信する方式である。送信関数より受信関数が先にコールされた場合には受信関数はデータの到着までブロック（停止）し、送信関数が先にコールされた場合には受信関数の発行までブロックするか、メッセージがシステム内にバッファリングされるのが一般的である。これはｓｅｎｄ／ｒｅｃｅｉｖｅ方式と呼ばれる。第２は、送信側ＰＵでメッセージの送信関数をコールしただけで（対応する受信関数の発行なしに）メッセージを相手ＰＵに書き込み、あるいは受信関数のコールにより（対応する送信関数の発行なしに）データを相手ＰＵから読み込む方式である。これはｐｕｔ／ｇｅｔ方式、あるいはリモートメモリコピー方式と呼ばれ、送信関数はｐｕｔ関数、受信関数はｇｅｔ関数と呼ばれる。
【０００５】上に挙げた標準的なメッセージパシングライブラリのうちＰＶＭではｓｅｎｄ／ｒｅｃｅｉｖｅ方式による通信のみをサポートしている。ＭＰＩもｓｅｎｄ／ｒｅｃｅｉｖｅ方式のみをサポートしているが、時期バージョンとして現在検討が進められているＭＰＩ－２ではこれに加えてｐｕｔ／ｇｅｔ方式による通信も取り入れる計画がある。
【０００６】ｓｅｎｄ／ｒｅｃｅｉｖｅ方式は、送信

側、受信側のＰＵがそれぞれに送信、受信関数を発行して明示的にメッセージを受け渡すため、利用者はどのタイミングでメッセージが自ＰＵのどのメモリ領域に送受信されるかを制御することができる。しかし、利用者はプログラム中で送信と受信の対応付けを常に行なわなくてはならないので、プログラムの手間は大きい。これに対し、ｐｕｔ／ｇｅｔ方式は送信側、受信側がそれぞれ一方的に相手ＰＵ上にあるデータを読み書きする。データが必要になった時点でｐｕｔ、あるいはｇｅｔの一方を発行すればよいので、プログラムの手間は少なくなる。しかし、メッセージの送受信先メモリ領域を誤った場合、あるいは２つ以上のＰＵがほぼ同時に同一のメモリ領域に読み書きを行なった場合には、データを破壊したり、データを上書きしたり、誤ったデータを読み出す恐れがある。

【０００７】ＭＰＩ－２では、送受信先メモリ領域の誤りにより利用者プログラムが破壊されることを防ぐ目的で、各ＰＵにｐｕｔ／ｇｅｔ専用のメモリ領域を確保する関数を提供している。ｐｕｔ／ｇｅｔ　操作は、各ＰＵであらかじめ確保された領域に対してしかメッセージの送受信を行なえないので、それ以外のメモリ領域に誤って書き出し、読み込みを行なう可能性がなくなる。

【０００８】以下、メッセージパシングライブラリＭＰＩ－２で提案されているｐｕｔ／ｇｅｔ関数を用いたメッセージ通信方法の概略を説明する。ただし、説明を簡単にするために、関数仕様は一部簡略化して記述する。

【０００９】ＭＰＩ利用時には、まず通信に参画する全プログラム上で次のようなＭＰＩ初期化関数をコールする。通常この処理は並列プログラムの先頭で行なう。

【００１０】Ｉｎｉｔ（）

本関数中で、メッセージパシングライブラリは必要な初期化操作を行なう。以下に挙げる関数は、初期化関数をコールした後でのみ使用することができる。

【００１１】初期か終了後ｐｕｔ／ｇｅｔを用いる際には、全ＰＵ上でｐｕｔ／ｇｅｔ用に専用のメモリ領域を確保するための次の関数をコールする。

【００１２】Ｒｍｃ＿ｍａｌｌｏｃ（ｂａｓｅ，　ｓｉｚｅ，　ｉｄ）

ここでｂａｓｅは確保するメモリ領域（ウィンドウ）の先頭アドレス、ｓｉｚｅはメモリ領域の長さである。ｉｄはウィンドウの識別子であり本関数の出力引き数である。本関数を複数回コールすることにより、ウィンドウを一つのＰＵ上に任意個数設けることができる。本関数で生成したウィンドウには、ｇｅｔ／ｐｕｔ関数からセットできるカウンタが自動的に付属する。このカウンタは直接利用者がアクセスすることはできないが、後述のハンドラを用いてその値を監視することで、ウィンドウに対するｐｕｔ／ｇｅｔ操作の完了を検知することができる。

【００１３】他のＰＵのウィンドウに対するｐｕｔ操作は次の関数コールにより実行する。

Ｐｕｔ（ｏｒｉｇｉｎ＿ａｄｄｒ，　ｏｒｉｇｉｎ＿ｓｉｚｅ，　ｔａｒｇｅｔ＿ｒａｎｋ，　ｉｄ，　ｉｎｃｒ）

ここでｏｒｉｇｉｎ＿ａｄｄｒ，　ｏｒｉｇｉｎ＿ｓｉｚｅは送るべきメッセージが格納された自ＰＵのメモリの先頭アドレスと長さである。ｔａｒｇｅｔ＿ｒａｎｋは送り先のＰＵ番号である。ｉｄはｐｕｔの対象となる相手ＰＵ上のウィンドウを示す。ｉｎｃｒは、当該ウィンドウに付属するカウンタのインクリメント値を表わし、メッセージの書き込みと同時にｐｕｔ先ウィンドウのカウンタがここに指定した値だけインクリメントされる。

【００１４】一方、ｇｅｔ操作は次の関数コールによる。

【００１５】Ｇｅｔ（ｏｒｉｇｉｎ＿ａｄｄｒ，　ｏｒｉｇｉｎ＿ｓｉｚｅ，　ｔａｒｇｅｔ＿ｒａｎｋ，　ｉｄ，　ｉｎｃｒ）

各引数の意味はｇｅｔの場合と同様である。ただし、本関数ではメッセージが相手ＰＵのウィンドウから読み出され、自ＰＵのｏｒｉｇｉｎ＿ａｄｄｒ以降のメモリに受信される。

【００１６】ｐｕｔ／ｇｅｔ操作の完了は、カウンタが利用者があらかじめ指定した上限値を越えたかどうかで判定する。ここで、カウンタの上限値は利用者がウィンドウごとに次の関数をコールすることにより設定することができる。

【００１７】Ｓｅｔ＿ｃｏｕｎｔｅｒ＿ｔｈｒｅｓｈｏｌｄ（ｉｄ，　ｃｏｕｎｔ）

ｉｄはウィンドウの識別子、ｃｏｕｎｔは設定すべき上限値である。カウンタが上限値を超えた場合には、利用者があらかじめ登録したハンドリング関数がＭＰＩによりコールバックされる。ハンドリング関数の設定は、次の関数コールにより行なう。

【００１８】

Ｐｏｓｔ＿ｈａｎｄｌｅｒ（ｉｄ，　ｈａｎｄｌｅｒ）

ｉｄはハンドリング関数の呼び出し対象となるウィンドウの識別子、ｈａｎｄｌｅｒはハンドリング関数へのポインタである。ハンドリング関数自体は利用者が自由に記述することができる。カウンタの上限値に１を指定し、ｐｕｔ／ｇｅｔの引き数ｉｎｃｒにも１を指定すれば、個々のｐｕｔ／ｇｅｔの終了時にハンドリング関数を呼び出すことができる。また、カウンタの上限値とｉｎｃｒで指定する値を適宜変更することで、複数回のｐｕｔ／ｇｅｔが終了した時点でハンドリング関数をコールすることも可能である。

【００１９】以上が、ＭＰＩを用いた場合の利用者プログラムからのｐｕｔ／ｇｅｔ操作方法の説明である。

【００２０】

【発明が解決しようとする課題】上述のようにｐｕｔ／

ｇｅｔ方式は潜在的にデータを破壊する可能性のある危険な通信方式であるにもかかわらず、ＭＰＩを含む従来のメッセージパシングライブラリでは２つ以上のＰＵが同一のメモリ領域に読み書きを行なった場合に対する考慮がなされていなかった。また、ｐｕｔ／ｇｅｔ操作によるデータの予期しない破壊が起こった場合に、利用者からこれを検出する手段が設けられていなかった。

【００２１】本発明の目的は、分散メモリ型並列計算機のメッセージパシングライブラリにおいて、利用者がｐｕｔ／ｇｅｔ操作による同一メモリ領域へのメッセージの重複した書き込みを防止することと、同一メモリ領域への重複した書き込みがあった場合にこれを検出することを可能とし、ｐｕｔ／ｇｅｔ方式による通信の安全性を高めることにある。

【００２２】

【課題を解決するための手段】本発明は、ｐｕｔ／ｇｅｔ方式による送受信おいて、複数のＰＵからの同一メモリ領域へのｐｕｔ／ｇｅｔの発行によるデータの上書きや誤ったデータの読み出しを防ぐために、各ＰＵ上でｐｕｔ／ｇｅｔ用に確保したメモリ領域に対して書き込み、読み出し禁止フラグを設け、そのフラグを利用者プログラムから制御可能とすることにより、利用者が当該メモリ領域への他ＰＵからの書き込み、読み出しを禁止するようにすることによって達成される。

【００２３】また、同一メモリ領域への重複した書き込み、読み出しを利用者が検出できるようにするために、書き込み、読み出し禁止フラグの立ったメモリ領域へのｐｕｔ／ｇｅｔの発行を利用者に通知するようにすることによって達成される。

【００２４】

【発明の実施の形態】以下、図を参照して本発明の詳細を説明する。

【００２５】まず、本発明の実装方法の具体例を図を参照して説明する。図１に本発明の全体構成図を示す。１０１，１０２はＰＵを示し、１０３，１０４はそれらのＣＰＵ、１０５，１０６はメモリである。１０７はそれらＰＵを結ぶ通信路である。ＰＵの数は実際には任意であるが、ここでは説明のために２つのＰＵからなる並列機を示している。１０８，１０９は各ＰＵのＯＳ（オペレーティングシステム）である。利用者プログラムを実行する際には、各ＰＵのメモリ上にプログラムがローディングされる（１１０，１１１）。利用者プログラムは、あらかじめ本発明のメッセージパシングライブラリ（１１４，１１５）とリンクされている。利用者プログラム中には、他のＰＵからリモートメモリ転送により読み書きが可能なメモリ領域（１２２，１２３）を設けることができる。さらに、並列プログラムの実行開始と同時に、ｐｕｔ／ｇｅｔデーモンプロセス（１１２，１１３）が各ＰＵごとに起動される。メッセージパシングライブラリ中には、ｐｕｔ／ｇｅｔデーモンからの割り込み要求によって起動される特別なルーチン（割り込みハンドラ）１２０，１２１およびｐｕｔ／ｇｅｔ用のメモリ領域の管理情報を格納するウィンドウ情報テーブル（１２４，１２５）が設けられる。割り込みハンドラ中には、ｐｕｔ要求にたいする割り込みを処理するｐｕｔハンドラと、ｇｅｔ要求に対する割り込みを処理するｇｅｔハンドラが存在する。

【００２６】以上の構成要素のうち、メッセージパシングライブラリとｐｕｔ／ｇｅｔデーモンとが本発明の特徴をなす構成要素である。以下、これら２つの構成要素について詳細に説明する。

【００２７】（メッセージパシングライブラリのインターフェイス）本発明におけるメッセージパシングライブラリ（１１４，１１５）について説明する。メッセージパシングライブラリは利用者の並列プログラムからコールすることにより、ｐｕｔ／ｇｅｔデーモンプロセスを介してメッセージパシングを実行するための関数群である。

【００２８】以下に、メッセージパシングライブラリの関数インターフェイスを説明する。本実施例では上述のＭＰＩ－２の仕様を基本とし、それに必要な変更を加えることによって関数インターフェイスを構築する。実際には関数名称、引き数名称などは任意であり、必ずしもここで説明する仕様と同じである必要はない。

【００２９】（１）Ｉｎｉｔ（）

本関数のインターフェイスは上述のＭＰＩと同様である。

【００３０】（２）ＲＭＣ＿Ｍａｌｌｏｃ（ｂａｓｅ，ｓｉｚｅ，ｃｏｕｎｔｅｒ，ｃｏｕｎｔｅｒ＿ｓｕｃｃｅｓｓ，ｉｄ）

自ＰＵ上にｐｕｔ／ｇｅｔウィンドウを確保する関数である。引き数ｂａｓｅ，ｓｉｚｅ，ｃｏｕｎｔｅｒ，ｉｄの意味は、従来の技術の項で述べたＭＰＩの場合と同様である。ｃｏｕｎｔｅｒ＿ｓｕｃｃｅｓｓは本発明にて新たに設けられたカウンタであり、利用法は後述する。

【００３１】（３）Ｇｅｔ（ｏｒｉｇｉｎ＿ａｄｄｒ，ｏｒｉｇｉｎ＿ｓｉｚｅ，ｔａｒｇｅｔ＿ｒａｎｋ，ｉｄ，ｉｎｃｒ）

本関数インターフェイスと引き数の意味は上述のＭＰＩの場合と同様である。

【００３２】（４）Ｐｕｔ（ｏｒｉｇｉｎ＿ａｄｄｒ，ｏｒｉｇｉｎ＿ｓｉｚｅ，ｔａｒｇｅｔ＿ｒａｎｋ，ｉｄ，ｉｎｃｒ）

本関数インターフェイスと引き数の意味は上述のＭＰＩの場合と同様である。

【００３３】（５）Ｓｅｔ＿ｃｏｕｎｔｅｒ＿ｔｈｒｅｓｈｏｌｄ（ｉｄ，ｃｏｕｎｔ）

本関数インターフェイスと引き数の意味は上述のＭＰＩの場合と同様である。

【0034】（6）Post_handler（id, count）
本関数インターフェイスと引き数の意味は上述のMPIの場合と同様である。
【0035】（7）Window_lock（id）
本関数と次のWindow_unlock関数は本発明に特徴的なものである。本関数が利用者プログラムから呼び出されると、引き数idで指定されたウィンドウ識別子に対応するウィンドウをput／get禁止とする。本操作をウィンドウのロックと呼ぶ。このウィンドウを対象とするput／get操作は、同じウィンドウに対して次のWindow_unlock関数がコールされるまで行なうことができない。ロックされたウィンドウに対してput操作を行なった場合には、そのメッセージは破棄される。ロックされたウィンドウにたいしてget操作を行なった場合には、メッセージは読みだされず、get関数の戻り値としてエラーコードが返される。
【0036】
（8）Window_unlock（id）
本関数は、idで指定されたウィンドウ識別子に対するウィンドウのput／get禁止を解除する。本操作をウィンドウのアンロックと呼ぶ。
【0037】（メッセージパシングライブラリの動作）
以下、本発明のリモートメモリへのデータ転送制御方式における上記各関数の実現方法を示す。
【0038】（1）Initの動作
メッセージパシングライブラリの初期化関数は、利用者プログラムからコールされると、図2に示すウィンドウ情報テーブルをライブラリのメモリ領域（図1の105，106）上に作成する。ウィンドウ情報テーブルは、ウィンドウ識別子のカラム、ウィンドウ開始アドレスのカラム、ウィンドウ末尾アドレスのカラム、カウンタ値のカラム、成功カウンタ値のカラム、カウンタの上限値のカラムおよびウィンドウのロック状態を示すフラグ（一般的にこの情報を保持できるものであればよい）のフィールドを有する。ついで、生成されたテーブルの各カラムを初期化する。初期値には、カウンタ上限値のカラムのみ1、他のすべてのカラムは0を用いる。上述のように利用者は初期化関数を全PUから呼び出すので、ウィンドウ情報テーブルも全PU上に作成される。
【0039】（2）Rmc_malloc関数の動作
ウィンドウ生成関数Rmc_mallocがコールされた場合の動作を図3に示す。まず、関数の引き数として与えられたウィンドウの先頭アドレスbuffと長さsizeを用いてウィンドウ用メモリを確保する（301）。ウィンドウ情報テーブルのidのカラムをスキャンし、idが0（初期値）であるカラムを見つける（302）。そのカラムの行番号をウィンドウの識別子として採用し、対応するカラムに格納する（303）。次いで、ウィンドウの先頭アドレスのカラムにbuffで与えられたアドレスを格納し、ウィンドウ末尾アドレスにbuff＋sizeの値を格納する（304，305）。
【0040】カウンタ値、カウンタ最大値、ロック状態フラグは初期値のまま変化させない。
【0041】（3）Put関数の動作
次に、Put関数がコールされた場合の動作を説明する。図4に処理の流れを示す。まず、相手PU上のput／getデーモンに対し、put要求が出されたことを通知する短いメッセージ（put要求）を送り、次いで関数の引き数として与えられた送信先ウィンドウの識別子とインクリメント値を、同じく引き数target_rank中で指定された相手PU上のput／getデーモンに送信する（401）。（put／getデーモンの動作は後述する。）この送信は、通常のsend関数を用いて行なう。送信後、同デーモンからの認識信号（ack）を待ち、ブロックする（402）。ackが到着したならば、同デーモンに対して、引き数のorigin_addrおよびorigin_sizeで示される送信バッファの内容を送信する（403）。この送信もsend関数によって行なう。送信バッファの内容がすべて送出されたならば、利用者プログラムへリターンする。
【0042】（4）Get関数の動作
同様に、Get関数がコールされた場合の動作を図5を用いて説明する。まず、まず、相手PU上のput／getデーモンに対し、put要求が出されたことを通知する短いメッセージ（get要求）を送り、次いで関数の引き数として与えられた送信先ウィンドウの識別子とインクリメント値を、同じく引き数target_rank中で指定された相手PU上のput／getデーモンに送信する（501）。この送信は、通常のsend関数を用いて行なう。送信後、同デーモンを対象としてreceive関数を発行しデータの到着を待つ。受け取ったメッセージは、引き数のorigin_addrおよびorigin_sizeで指定された受信バッファに格納する（502）。受信バッファへ内容がすべて格納されたならば、利用者プログラムへリターンする。
【0043】（5）Set_counter_threshold関数の動作
本関数がコールされると、自PUのウィンドウ情報テーブル（図2）から引き数で指定された識別子を持つ行をサーチし、その行のカウンタ上限値のカラムに引き数で指定されたカウンタ上限値を格納する。
【0044】（6）Post_hander関数の動作
本関数がコールされると、自PUのウィンドウ情報テーブル（図2）から引き数で指定された識別子を持つ行をサーチし、その行のハンドラのカラムに引き数で指定されたハンドリング関数のポインタを格納する。

【００４５】（７）Ｗｉｎｄｏｗ＿ｌｏｃｋ関数の動作
本関数がコールされると、自ＰＵのウィンドウ情報テーブル（図２）から引き数で指定された識別子を持つ行をサーチし、その行のロック状態フラグ（書き込み、読み出し禁止フラグ）をｏｎにする。
【００４６】（８）Ｗｉｎｄｏｗ＿ｕｎｌｏｃｋ関数の動作
本関数がコールされると、自ＰＵのウィンドウ情報テーブル（図２）から引き数で指定された識別子を持つ行をサーチし、その行のロック状態フラグ（書き込み、読み出し禁止フラグ）をｏｆｆにする。
【００４７】以上が本発明のリモートメモリ転送制御方式における各関数の実現方法の例である。
【００４８】（９）ｐｕｔハンドラの動作
メッセージパシングライブラリ中には、利用者からコールされる上記の関数群以外に、ｐｕｔ／ｇｅｔデーモンプロセスからの割り込みによって起動されるｐｕｔ／ｇｅｔハンドラが存在する。このうちｐｕｔハンドラの動作を図７を参照して説明する。割り込みを受け付けると、ｐｕｔ／ｇｅｔデーモンプロセスからの受信待ち状態に移行し、ウィンドウ識別子とカウンタのインクリメント値を受け取る（７０１）。識別子を受け取ると、それを用いて自ＰＵ上のウィンドウ情報テーブル（図２）をサーチし、対応するカウンタを受け取ったインクリメント値にしたがってインクリメントする。また、ロックフラグのカラムを調べて当該ウィンドウがロック状態にあるか否かを調べる（７０４）。ロックされていない場合には、ｐｕｔ／ｇｅｔデーモンプロセスにａｃｋを返し（７０２）、続いてｐｕｔ／ｇｅｔデーモンプロセスから送信されてくるメッセージ本体を受け取り、ウィンドウに格納する（７０３）。また、対応する成功カウンタの値をインクリメント値にしたがってインクリメントする（７０６）。ロックされていた場合にはｐｕｔ／ｇｅｔデーモンプロセスにｎａｃｋを返す（７０５）。最後にカウンタ値とカウンタ上限値とを比較し、カウンタ値が等しいか大きくなっていればカウンタをクリアして対応する利用者定義のハンドラに制御を移す（７０７）。
（１０）ｇｅｔハンドラの動作
ｇｅｔハンドラの動作を図８を参照して説明する。割り込みを受け付けると、ｐｕｔ／ｇｅｔデーモンプロセスからの受信待ち状態に移行し、ウィンドウ識別子を受け取る（８０１）。識別子を受け取ると、それを用いて自ＰＵ上のウィンドウ情報テーブル（図２）をサーチし、ロックフラグのカラムを調べて当該ウィンドウがロック状態にあるか否かを調べる（８０４）。ロックされていない場合には、ｐｕｔ／ｇｅｔデーモンプロセスにａｃｋを返し（８０２）、続いてウィンドウに格納されているメッセージ本体をｐｕｔ／ｇｅｔデーモンプロセスに送信する（８０３）。ロックされていた場合にはｐｕｔ／ｇｅｔデーモンプロセスにｎａｃｋを返す（８０５）。最後にカウンタ値とカウンタ上限値とを比較し、カウンタ値が等しいか大きくなっていればカウンタをクリアして対応する利用者定義のハンドラに制御を移す（８０７）。
【００４９】（１１）ロックされているウィンドウに対するｐｕｔ／ｇｅｔの検出
本発明では、何等かの原因でロックされているウィンドウにメッセージがｐｕｔ／ｇｅｔされた場合、これを次のようにして検出することができる。
【００５０】ｇｅｔ関数に対しては、ｇｅｔ先のｇｅｔハンドラからｎａｃｋが返された場合、ｐｕｔ／ｇｅｔデーモンプロセスはｇｅｔ関数発行元にたいしてエラーコードを返す（６１９）。したがって、利用者はｇｅｔ関数のリターン値がエラーコードであった場合には、ロックされたウィンドウに対してｇｅｔが発行されたことを検知できる。
【００５１】ｐｕｔ関数は、相手ＰＵに対してメッセージを送信した後直ちにリターンするので、リターンコードによってロックされたウィンドウに書き込みを行なおうとしたことを検出できない。しかし、ｐｕｔハンドラはｐｕｔが成功した場合には成功カウンタとカウンタの双方をインクリメントするのに対して、ｐｕｔが成功しなかった場合には成功カウンタをインクリメントしない。したがって、利用者はｐｕｔ関数の完了ハンドリング関数中で両者の値を比較することにより、ロックしたウィンドウへのｐｕｔが行われたことを検知できる。すなわち、ロックされたウィンドウへｐｕｔが行われた場合には、成功カウンタの値がカウンタの値よりも小さくなっているはずである。
【００５２】（ｐｕｔ／ｇｅｔデーモンの実現方法）次に、本発明のリモートメモリ転送方式におけるｐｕｔ／ｇｅｔデーモンの実現方法の例を図６を参照して説明する。
【００５３】ｐｕｔ／ｇｅｔデーモンは通常はいずれかのＰＵのユーザプログラムからのｐｕｔ／ｇｅｔ要求待ちでブロック状態にある（６０１）。ｐｕｔ／ｇｅｔ関数コールによりｐｕｔ／ｇｅｔ要求を受信すると、それがｐｕｔ要求であるかｇｅｔ要求であるかによって分岐し（６０２）、各々に対応する処理に入る。ｐｕｔ要求であった場合には、まず送信されてくるウィンドウ識別子とインクリメント値を読み込んで記憶し（６０３）、ｐｕｔ要求元プログラムにａｃｋを返す（６０４）。次いで、ｐｕｔ要求元から送信されてくるメッセージ本体を受信して記憶し（６０５）、メッセージをすべて受診すると自ＰＵ上の利用者プログラムにたいして割り込みを発生する（６０６）。（この割り込みによって利用者プログラム側ではｐｕｔハンドラがコールされ、受信待ち状態となる。）デーモンは利用者プログラムに対して、６０３にて記憶したウィンドウ識別子を送信し、肯

11

定応答（ａｃｋ）または否定応答（ｎａｃｋ）を待って処理を分岐する（６１０）。ａｃｋを受け取った場合には、当該ウィンドウはロック状態にはないので、６０５にて記憶したメッセージ本体を利用者プログラムに送信して処理を終了する。ｎａｃｋを受け取った場合には、当該ウィンドウはロックされているので、メッセージを破棄して処理を終了する。一方６０２にてｇｅｔ要求へ分岐した場合には、要求元ＰＵからウィンドウ識別子を受け取り（６１２）、自ＰＵ上の利用者プログラムに割り込みをかける（６１３）。（この割り込みによって利用者プログラム側ではｇｅｔハンドラがコールされ、受信待ち状態となる。）デーモンは利用者プログラムに対して、６１２にて記憶したウィンドウ識別子を送信し、肯定応答（ａｃｋ）または否定応答（ｎａｃｋ）を待って処理を分岐する（６１８）。ａｃｋを受け取った場合には、当該ウィンドウはロック状態にはないので、利用者プログラムのｇｅｔ処理ハンドリング関数よりメッセージが送られてくる。これを記憶し、ｇｅｔ要求元に送信して処理を終了する（６１６，６１７）。ｎａｃｋを受け取った場合には、当該ウィンドウはロックされておりメッセージは送られてこないので、ｇｅｔ要求元にエラーコードを返す（６１９）。

【００５４】以上がｐｕｔ／ｇｅｔデーモンの実現方法の例である。

【００５５】

【発明の効果】本発明のリモートメモリ転送制御方式によれば、メッセージパシングライブラリを用いたｐｕｔ／ｇｅｔ方式のメッセージ通信において、ｐｕｔ／ｇｅｔ用のメモリ領域を利用者プログラム中からロック、アンロックすることが可能となる。これによって利用者はｐｕｔ／ｇｅｔ用のメモリ領域に複数のＰＵからのデータが上書きされたり、不正なデータが読みだされたりす＊

12

＊ることを防止することができ、ｐｕｔ／ｇｅｔ方式によるメッセージ通信をより安全に行なうことができるようになる。

【００５６】また、本発明のリモートメモリ転送制御方式によれば、何等かの原因でロックされたメモリ領域にたいして誤ったｐｕｔ／ｇｅｔがなされた場合に、これを検出することができる。これによって利用者は、適当なエラーリカバリをプログラム中に組み込むことが可能となり、ｐｕｔ／ｇｅｔ方式を用いて信頼性の高いプログラムを作成することができる。

【図面の簡単な説明】

【図１】本発明の実施例の全体構成図。

【図２】ウィンドウ情報テーブルの説明図。

【図３】ウィンドウ生成関数の動作説明図。

【図４】ｐｕｔ関数の動作説明図。

【図５】ｇｅｔ関数の動作説明図。

【図６】ｐｕｔ／ｇｅｔデーモンプロセスの動作説明図。

【図７】ｐｕｔハンドラの動作説明図。

【図８】ｇｅｔハンドラの動作説明図。

【符号の説明】

１０１，１０２．．．要素計算機
１０３，１０４．．．ＣＰＵ
１０５，１０６．．．メモリ
１０７．．．通信路
１０８，１０９．．．オペレーティングシステム
１１０，１１１．．．利用者プログラム
１１２，１１３．．．ｐｕｔ／ｇｅｔデーモンプロセス
１１４，１１５．．．メッセージパシングライブラリ
１２０，１２１．．．ｐｕｔ／ｇｅｔ割り込みハンドラ
１２２，１２３．．．ｐｕｔ／ｇｅｔ用ウィンドウ。

【図２】

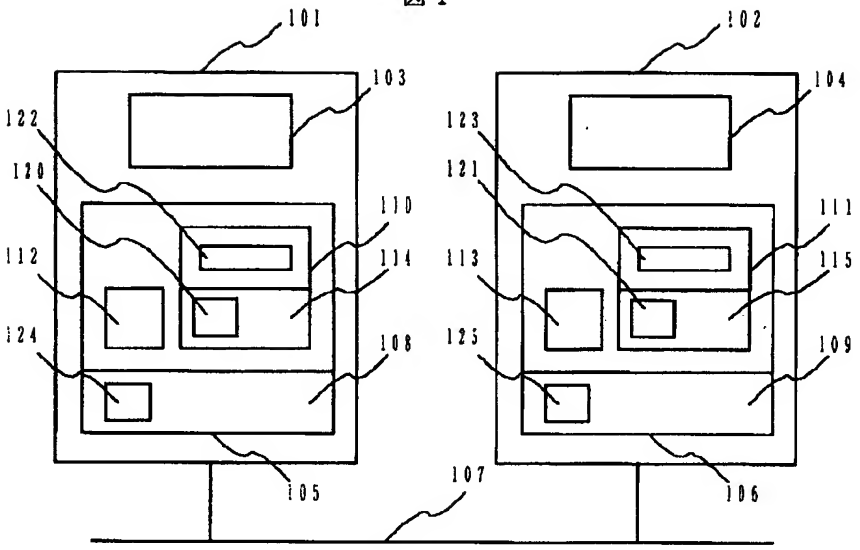
図２

101

ウィンドウ識別子	バッファ先頭アドレス	バッファ末尾アドレス	カウンタ値	成功カウンタ値	カウンタ上限値	ロック状態フラグ	ハンドラ

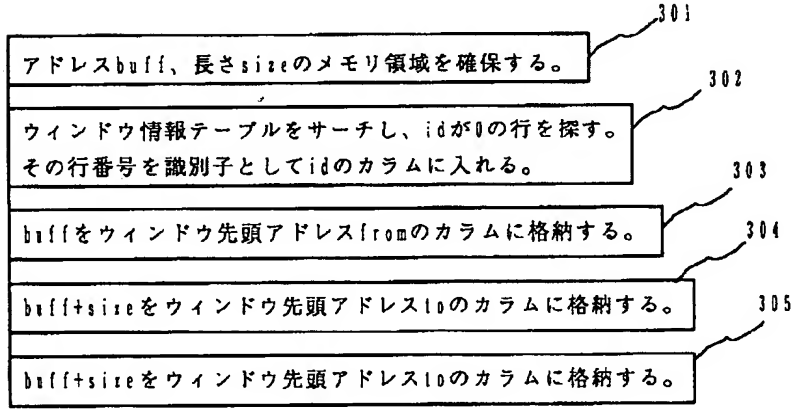
【図1】

図1


101
102
103
104
122
123
120
121
110
111
112
113
114
115
124
125
108
109
105
106
107

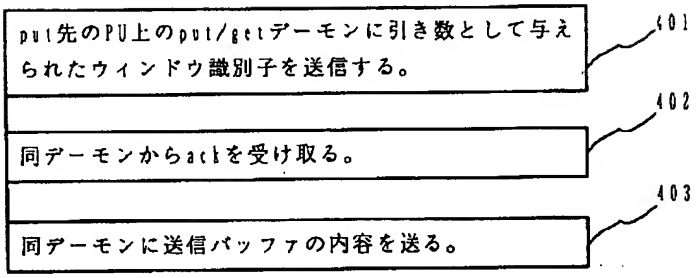
【図3】

図3


301
アドレスbuff、長さsizeのメモリ領域を確保する。
302
ウィンドウ情報テーブルをサーチし、idが0の行を探す。
その行番号を識別子としてidのカラムに入れる。
303
buffをウィンドウ先頭アドレスfromのカラムに格納する。
304
buff+sizeをウィンドウ先頭アドレスtoのカラムに格納する。
305
buff+sizeをウィンドウ先頭アドレスtoのカラムに格納する。

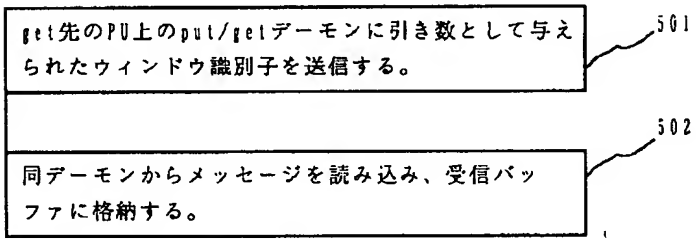
【図4】

図4



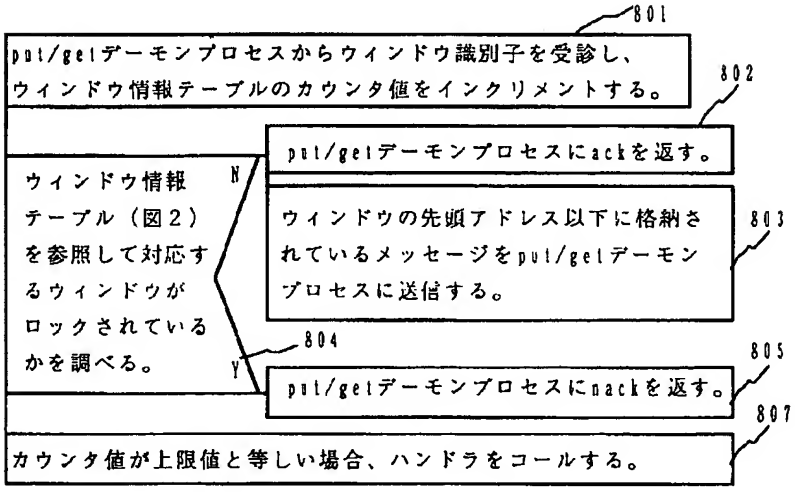
【図5】

図5



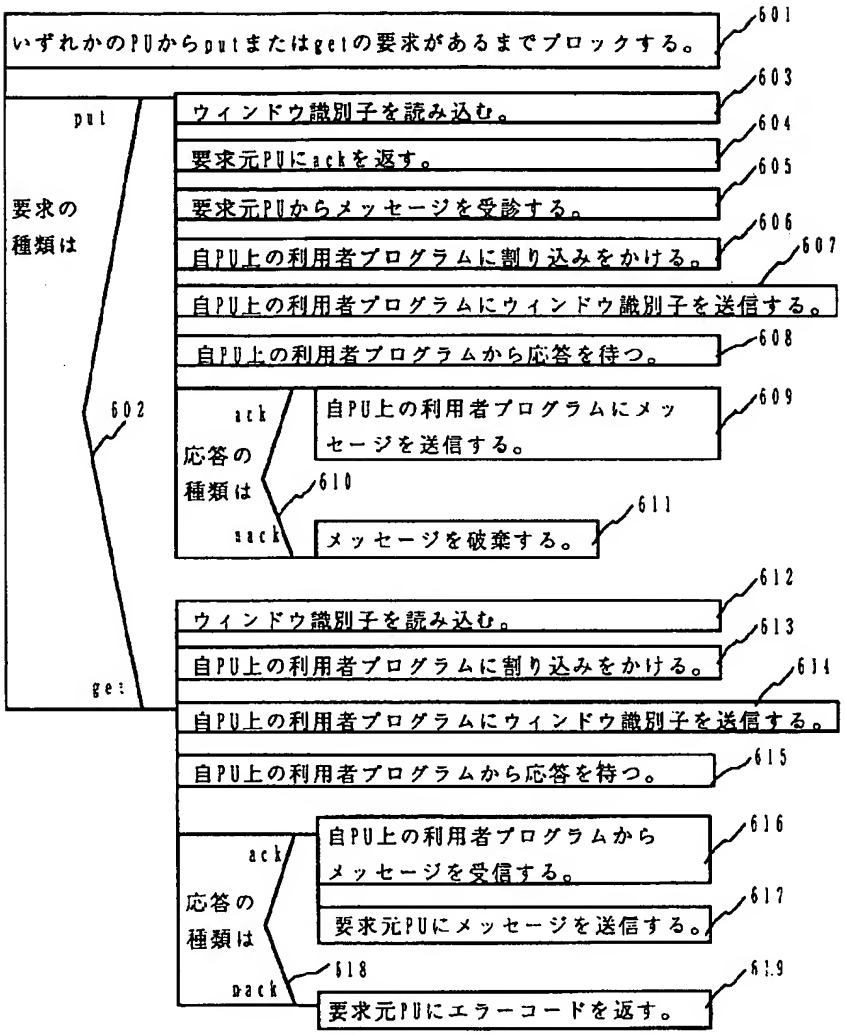
【図8】

図8



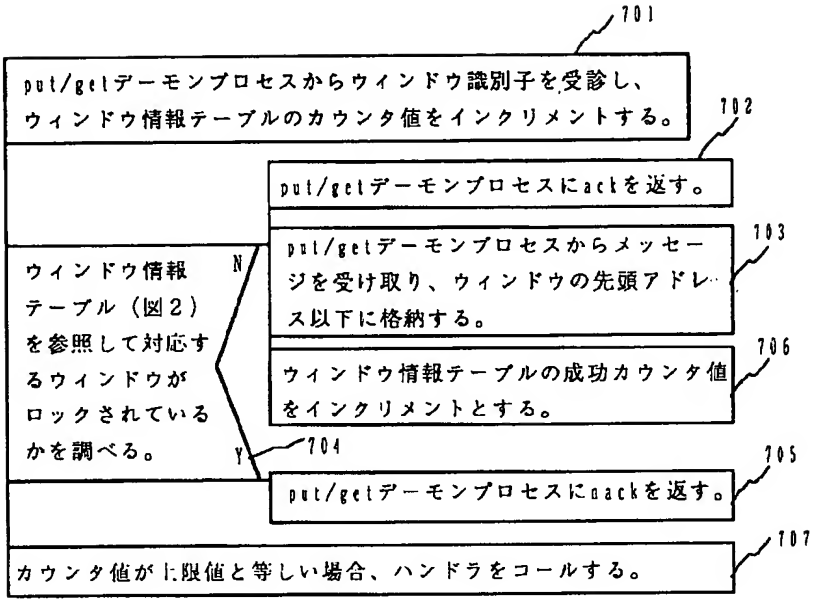
【図6】

図6


いずれかのPUからputまたはgetの要求があるまでブロックする。
601
要求の種類は
602
put
ウィンドウ識別子を読み込む。
603
要求元PUにackを返す。
604
要求元PUからメッセージを受診する。
605
自PU上の利用者プログラムに割り込みをかける。
606
自PU上の利用者プログラムにウィンドウ識別子を送信する。
607
自PU上の利用者プログラムから応答を待つ。
608
応答の種類は
ack
自PU上の利用者プログラムにメッセージを送信する。
609
nack
610
メッセージを破棄する。
611
get
ウィンドウ識別子を読み込む。
612
自PU上の利用者プログラムに割り込みをかける。
613
自PU上の利用者プログラムにウィンドウ識別子を送信する。
614
自PU上の利用者プログラムから応答を待つ。
615
応答の種類は
ack
自PU上の利用者プログラムからメッセージを受信する。
616
要求元PUにメッセージを送信する。
617
nack
618
要求元PUにエラーコードを返す。
619

【図7】

図7


701
put/getデーモンプロセスからウィンドウ識別子を受診し、ウィンドウ情報テーブルのカウンタ値をインクリメントする。
702
put/getデーモンプロセスにackを返す。
ウィンドウ情報テーブル（図2）を参照して対応するウィンドウがロックされているかを調べる。
N
703
put/getデーモンプロセスからメッセージを受け取り、ウィンドウの先頭アドレス以下に格納する。
706
ウィンドウ情報テーブルの成功カウンタ値をインクリメントとする。
Y
704
705
put/getデーモンプロセスにnackを返す。
707
カウンタ値が上限値と等しい場合、ハンドラをコールする。